

WEST Search History

DATE: Thursday, October 23, 2003

<u>Set Name</u>	<u>Query</u>	<u>Hit Count</u>	<u>Set Name</u>
side by side			result set
	<i>DB=TDBD; PLUR=YES; OP=ADJ</i>		
L19	L8	1	L19
	<i>DB=USPT,PGPB; PLUR=YES; OP=ADJ</i>		
L18	L8 and l1	11	L18
L17	L8 and l4	22	L17
	<i>DB=EPAB,DWPI; PLUR=YES; OP=ADJ</i>		
L16	L15	18	L16
	<i>DB=JPAB,EPAB,DWPI; PLUR=YES; OP=ADJ</i>		
L15	L7	20	L15
	<i>DB=EPAB; PLUR=YES; OP=ADJ</i>		
L14	L7	2	L14
L13	L8	0	L13
L12	L11	0	L12
	<i>DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ</i>		
L11	L10 and (byte?code or bytecode)	19	L11
L10	L8 and redundant	22	L10
L9	L8 and portal	8	L9
L8	L7 and constant near pool	121	L8
L7	(link\$ or map\$) and class file	1204	L7
	<i>DB=USPT,PGPB; PLUR=YES; OP=ADJ</i>		
L6	L5 or l4 or l3 or l2 or l1	27191	L6
L5	((712/2)!.CCLS.)	73	L5
	((709/200 709/201 709/202 709/203 709/204 709/205 709/206 709/207 709/208 709/209 709/210 709/211 709/212 709/213 709/214 709/215 709/216 709/217 709/218 709/219 709/220 709/221 709/222 709/223 709/224 709/225 709/226 709/227 709/228 709/229 709/230 709/231 709/232 709/233 709/234 709/235 709/236 709/237 709/238 709/239 709/240 709/241 709/242 709/243 709/244 709/245 709/246 709/247 709/248 709/249 709/250 709/251 709/252 709/253 709/310 709/311 709/312 709/313 709/314 709/315 709/331)!.CCLS.)	25210	L4
L4			
L3	((435/804 435/805)!.CCLS.)	955	L3
L2	((717/100 717/107 717/108 717/116 717/118)!.CCLS.)	769	L2
L1	((717/162 717/163 717/164 717/165 717/166 717/167)!.CCLS.)	424	L1

Generate Collection

Print

IBM Tech Bulletin *Revised* Search Results - Record(s) 1 through 1 of 1 returned.

☐ 1. Document ID: NNRD453168

L19: Entry 1 of 1

File: TDBD

Jan 1, 2002

TDB-ACC-NO: NNRD453168

DISCLOSURE TITLE: Improved field layout of Java classes

PUBLICATION-DATA:

IBM technical Disclosure Bulletin, January 2002, UK

ISSUE NUMBER: 453

PAGE NUMBER: 150

DISCLOSURE TEXT:

A program is disclosed to efficiently lay out fields in a Java* Virtual Machine, particularly on 64-bit systems. Java objects have fields, which can hold data of various sizes. An object can hold fields of the following sizes: Type Minimum size Typical size Typical alignment boolean 1 bit 4 bytes 4 bytes byte 1 byte 4 bytes 4 bytes char or short 2 bytes 4 bytes 4 bytes int 4 bytes 4 bytes 4 bytes long 8 bytes 8 bytes 4 or 8 bytes float 4 bytes 4 bytes 4 bytes double 8 bytes 8 bytes 4 or 8 bytes reference natural pointer size 4 or 8 bytes 4 or 8 bytes

SECURITY: Use, copying and distribution of this data is subject to the restrictions in the Agreement For IBM TDB Database and Related Computer Databases. Unpublished - all rights reserved under the Copyright Laws of the United States. Contains confidential commercial information of IBM exempt from FOIA disclosure per 5 U.S.C. 552(b)(4) and protected under the Trade Secrets Act, 18 U.S.C. 1905.

COPYRIGHT STATEMENT: The text of this article is Copyrighted (c) IBM Corporation 2002. All rights reserved.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

FOUO	Draw Desc
------	-----------

Generate Collection

Print

Terms	Documents
L8	1

Display Format:

REV

Change Format

[Previous Page](#)

[Next Page](#)

Generate Collection

Print

Dewent

JPO, EPO, record

Search Results - Record(s) 1 through 18 of 18 returned.

☐ 1. Document ID: WO 2086702 A1

L16: Entry 1 of 18

File: EPAB

Oct 31, 2002

PUB-NO: WO002086702A1

DOCUMENT-IDENTIFIER: WO 2086702 A1

TITLE: HANDLING DIFFERENT SERVICE VERSIONS IN A SERVER

PUBN-DATE: October 31, 2002

INVENTOR-INFORMATION:

NAME

KROHN, PETRI

JAESKE, HARRI

COUNTRY

FI

FI

INT-CL (IPC): G06 F 9/44

EUR-CL (EPC): G06F009/44; G06F009/44

ABSTRACT:

CHG DATE=20030114 STATUS=O>This invention relates to handling of different service versions in a server that is connected to a communication network. The invention comprises means to load a desired version, two tables and additional data for handling different service versions. The first table contains service key and version information, and serialized service objects. The second table contains names of classes, version information and class files. The additional data is needed for loading the right class from among classes, with the same name, and mapping the right service object version to the right versions of classes.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

FWMC	Draw Desc	Image
------	-----------	-------

☐ 2. Document ID: EP 1128263 A2

L16: Entry 2 of 18

File: EPAB

Aug 29, 2001

PUB-NO: EP001128263A2

DOCUMENT-IDENTIFIER: EP 1128263 A2

TITLE: Program generation apparatus

PUBN-DATE: August 29, 2001

INVENTOR-INFORMATION:

NAME

KAWAI, MASAKI

KAWAMOTO, TAKUJI

COUNTRY

JP

JP

INT-CL (IPC): G06 F 9/44

EUR-CL (EPC): G06F009/44; G06F009/44

ABSTRACT:

CHG DATE=20011002 STA S=O> A program generation apparatus for generating lightweight class files for each terminal apparatus by linking class files. The program generation apparatus includes: a storage unit for prestoring the class files for each terminal apparatus, where each class defines (a) dependent variables unique to each terminal apparatus and (b) non-dependent variables common to all the terminal apparatuses, each variable is identified by a variable name, and each class file includes a variable name for each variable; an assignment unit for assigning an offset number to each variable defined in the class files so that the same offset numbers are assigned to non-dependent variables having the same variable names; and a generation unit for generating the lightweight class files for each terminal apparatus by replacing each variable name in each class file with an offset number assigned by the assignment unit.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

WAC	Draw Desc	Image
-----	-----------	-------

☐ 3. Document ID: US 20030093401 A1

L16: Entry 3 of 18

File: DWPI

May 15, 2003

DERWENT-ACC-NO: 2003-576815

DERWENT-WEEK: 200354

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Meta data sharing facilitation method e.g. for class files, involves referring object in shared library using corresponding symbolic names

INVENTOR: CZAJKOWSKI, G J; DAYNES, L P

PRIORITY-DATA: 2001US-332924P (November 14, 2001), 2002US-0102318 (March 19, 2002)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 20030093401 A1	May 15, 2003		011	G06F007/00

INT-CL (IPC): G06 F 7/00

ABSTRACTED-PUB-NO: US20030093401A

BASIC-ABSTRACT:

NOVELTY - A shared library (116) containing the set of objects (C1-C3) defining class, is generated. A symbol table (202) is configured within the library to include an identifier for each object. The shared library is accessed through a set of symbolic names (LibC1-LibC3). Each object on the library is referred using corresponding symbolic name, to refer the object.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

- (1) recorded medium storing the sharing objects facilitation program;
- (2) sharing object facilitation apparatus; and
- (3) computer system.

USE - For facilitating sharing of meta data such as class files or dynamically compiled code by instances of running system such as Java virtual machine (JVM) for use in computer systems such as microprocessor, main print computer, digital signal processor, personal organizer and portable computing device.

ADVANTAGE - By storing multiple meta data objects in the same shared library, the internal fragmentation that arises when page sizes relatively large, is avoided. Avoids maintaining a mapping from meta data object names to library names, and facilitate referring a shared library containing object without complication.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of the shared library in the computer system.

shared library 116

symbol table 202

objects C1-C3

symbolic names LibC1-LibC3

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

KWMC	Draw Desc	Clip Img	Image
------	-----------	----------	-------

☐ 4. Document ID: US 20030088851 A1

L16: Entry 4 of 18

File: DWPI

May 8, 2003

DERWENT-ACC-NO: 2003-567696

DERWENT-WEEK: 200353

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Global constant management method for computer memory, involves storing data structure and its constant in different structures in memory, and replacing constants with links to respective other data structures in memory

INVENTOR: HARSCOET, P

PRIORITY-DATA: 1999US-0347037 (July 2, 1999), 2002US-0327482 (December 19, 2002)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 20030088851 A1	May 8, 2003		010	G06F009/44

INT-CL (IPC): G06 F 9/44

ABSTRACTED-PUB-NO: US20030088851A

BASIC-ABSTRACT:

NOVELTY - The method involves receiving a data structure from a memory, and storing the data structure in another memory. The constants in the data structures are stored in some other data structure in the latter memory. The constants from the received data structure in the latter memory are then replaced with links to respective other data structures in the latter memory.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for a computer system.

USE - Used for management of global constants in a computer memory.

ADVANTAGE - The method avoids redundancies between commonly used constants from their respective classes. The storing of the constants in objects separately from the class files enables replacement of commonly repeated portions of constants by shorter codes.

DESCRIPTION OF DRAWING(S) - The drawing shows a flow chart of loading classes into memory.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

KWMC	Draw Desc	Clip Img	Image
------	-----------	----------	-------

☐ 5. Document ID: WO 200297619 A2

L16: Entry 5 of 18

File: DWPI

Dec 5, 2002

DERWENT-ACC-NO: 2003-321
DERWENT-WEEK: 200305
COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Mobile communication device runtime process system has pre-linked module that has classes which are loaded and linked, and has information regarding closed set loaded and linked class files

INVENTOR: DAHMS, J F A; YACH, D P

PRIORITY-DATA: 2001US-294331P (May 30, 2001)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
WO 200297619 A2	December 5, 2002	E	036	G06F009/445

INT-CL (IPC): G06 F 9/445

ABSTRACTED-PUB-NO: WO 200297619A
BASIC-ABSTRACT:

NOVELTY - A pre-linked module has classes which are loaded and linked, and has information regarding closed set loaded and linked class files. The pre-linked module is accessed during execution of the application.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are included for the following:

- (1) Communication system;
- (2) Classes host-linking method;
- (3) Host-linked module; and
- (4) Computer data signal.

USE - For splitting processing machine runtime between host system and target system.

ADVANTAGE - The need for application to repeat loading and linking of classes is eliminated during execution of application, due to use of pre-linked module. Allows runtime to be efficiently split between host system and target system to optimize runtime efficiency at target system. The pre-linked module has information including closed set loaded and linked class files, thereby optimizing commands, symbolic information and code size and speed target processor.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of runtime split system.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

FullC	Draw Desc	Clip Img	Image
-------	-----------	----------	-------

☐ 6. Document ID: US 20020059475 A1 EP 1207454 A1

L16: Entry 6 of 18

File: DWPI

May 16, 2002

DERWENT-ACC-NO: 2002-489512
DERWENT-WEEK: 200252
COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Java run-time system for embedded microcontrollers of smart card, has converter to map standard Java symbolic linking strings in Java program onto linking identifiers

INVENTOR: BAENTSCH, M; BUHLER, P ; EIRICH, T ; HOERING, F ; OESTREICHER, M ;
WEIGOLD, T D

PRIORITY-DATA: 2000EP 11077 (November 15, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 20020059475 A1	May 16, 2002		010	G06F009/00
EP 1207454 A1	May 22, 2002	E	000	G06F009/445

INT-CL (IPC): G06 F 9/00; G06 F 9/445

ABSTRACTED-PUB-NO: US20020059475A

BASIC-ABSTRACT:

NOVELTY - A stacked-based interpreter executes a Java program comprising Java bytecode instructions and class structures. A converter (100) maps the standard Java symbolic linking strings (80) contained in the Java program onto linking identifiers (65). An export table (40) including the linking identifiers is framed to bind a reference in the bytecode instruction to be executed to a corresponding link target (90).

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are included for the following:

- (1) Java development kit;
- (2) Java program downloading and linking method; and
- (3) Memory device storing Java program downloading and linking program.

USE - For embedded microcontrollers of smart cards such as public telephone chip cards.

ADVANTAGE - The amount of memory space on embedded microcontrollers of chip cards are saved by effective mapping of long Java symbolic linking strings onto short identifiers. Mapping of different symbolic linking strings to the same token is avoided. Thereby, the conversion of the standard Java class file into Java card cap file is made easy.

DESCRIPTION OF DRAWING(S) - The figure shows the schematic view illustrating the downloading and linking performance adopted in the Java card run-time system.

Export table 40

Identifiers 65

Java symbolic linking strings 80

Link target 90

Converter 100

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

FIG. 1	Draw Desc	Clip Img	Image
--------	-----------	----------	-------

☐ 7. Document ID: AU 200211609 A WO 200237272 A2

L16: Entry 7 of 18

File: DWPI

May 15, 2002

DERWENT-ACC-NO: 2002-426784

DERWENT-WEEK: 200258

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Internal class representation data structure in computer readable medium for use by virtual machine at runtime comprises method and reference cell corresponding to method

INVENTOR: SOKOLOV, S

PRIORITY-DATA: 2000US 03361 (October 31, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
AU 200211609 A	May 15, 2002		000	G06F009/44
WO 200237272 A2	May 10, 2002	E	026	G06F009/44

INT-CL (IPC): G06 F 9/44

ABSTRACTED-PUB-NO: WO 200237272A

BASIC-ABSTRACT:

NOVELTY - The representation data structure includes a first method and a reference cell that corresponds to the method. The reference cell includes a class pointer field that can be used to locate an internal representation of a class. A method name field contains or references the name of the first method. A signature field contains or references a signature associated with the method.

DETAILED DESCRIPTION - The reference cell further includes an information field for containing or referencing information generated at runtime by the virtual machine and a link field. The link field contains information suitable for directly or indirectly linking the reference cell to the internal class representation.

INDEPENDENT CLAIMS are included for a process of loading a class files into a computing system.

USE - For invoking methods in virtual computing machines.

ADVANTAGE - Provides improved frameworks for invoking methods in virtual machines e.g. Java virtual machines.

DESCRIPTION OF DRAWING(S) - The figure shows an internal class representation.

Full Title Citation Front Review Classification Date Reference Sequences Attachments

WMC Draw Desc Clip Img Image

☐ 8. Document ID: EP 1174796 A2 JP 2002099449 A

L16: Entry 8 of 18

File: DWPI

Jan 23, 2002

DERWENT-ACC-NO: 2002-165965

DERWENT-WEEK: 200239

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Application class file instrumentation in data processing system, involves modifying class file by updating static initializer to output specific hook that maps routine name to major and minor codes

INVENTOR: BERRY, R F; GU, W ; HUSSAIN, R Y ; LEVINE, F E ; WONG, W Y P

PRIORITY-DATA: 2000US-0620729 (July 20, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
EP 1174796 A2	January 23, 2002	E	013	G06F011/34
JP 2002099449 A	April 5, 2002		014	G06F011/34

INT-CL (IPC): G06 F 9/44; G06 F 11/34; G06 F 11/36

ABSTRACTED-PUB-NO: EP 1174796A

BASIC-ABSTRACT:

NOVELTY - A class file is instrumented with hooks that contain unique major and minor codes, to identify a routine. The class file is modified by updating a static initializer in the class, so as to output another hook that identifies the

instrumented routine mapping the routing name, to the major and minor codes.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

- (a) Data processing system for supporting user-specified instrumentation;
- (b) Computer program product for supporting user-specified instrumentation

USE - For instrumenting class files of Java applications or programs running on data processing systems such as symmetric multiprocessor system e.g. IBM RISC/System 6000 system. Also for implementation in personal computers, workstations, embedded systems, mini computers and main frame computers.

ADVANTAGE - Provides correlation information between class files and instrumentation, by using hooks that map the routine names to major and minor codes in the class. Hence, the process for generating post process profiling information report with method names, is much more simplified, without the need for users to find and validate the hook definition file (HDF).

DESCRIPTION OF DRAWING(S) - The figure shows the flowchart explaining post-process report generation with method names.

Full Title Citation Front Review Classification Date Reference Sequences Attachments

RMK Draw Desc Clip Img Image

☐ 9. Document ID: JP 2001256058 A US 20010037495 A1

L16: Entry 9 of 18

File: DWPI

Sep 21, 2001

DERWENT-ACC-NO: 2001-643315

DERWENT-WEEK: 200212

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Interpreter language program execution method in information processing system, involves specifying memory access tip based on reference solution result when memory access is to be executed for a program

INVENTOR: BABA, Y; KATO, M ; NAKAGAWA, S ; YANAGI, H

PRIORITY-DATA: 2000JP-0069113 (March 13, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
JP 2001256058 A	September 21, 2001		017	G06F009/45
US 20010037495 A1	November 1, 2001		000	G06F009/45

INT-CL (IPC): G06 F 9/45

ABSTRACTED-PUB-NO: JP2001256058A

BASIC-ABSTRACT:

NOVELTY - A reference information specifying memory access tip for interpreter language program execution, is extracted and solved. When a program which requires memory access to be executed, the memory access tip is specified based on reference solution result linked to the program through the saved reference information.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for information processor.

USE - In information processing system.

ADVANTAGE - By using the saved reference solution result, it is possible to quickly access the memory, thereby achieving stability and speed of program execution. Since execution time of program becomes fixed, it is possible to provide execution time of program for real-time processing which require them. The program need not be rewritten depending on reference solution, thereby reduction in RAM capacity is

enabled.

DESCRIPTION OF DRAWING(S) - The figure shows the relationship between instruction component and class file information and relationship between the two. (Drawing includes non-English language text).
ABSTRACTED-PUB-NO:

US20010037495A EQUIVALENT-ABSTRACTS:

NOVELTY - A reference information specifying memory access tip for interpreter language program execution, is extracted and solved. When a program which requires memory access to be executed, the memory access tip is specified based on reference solution result linked to the program through the saved reference information.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for information processor.

USE - In information processing system.

ADVANTAGE - By using the saved reference solution result, it is possible to quickly access the memory, thereby achieving stability and speed of program execution. Since execution time of program becomes fixed, it is possible to provide execution time of program for real-time processing which require them. The program need not be rewritten depending on reference solution, thereby reduction in RAM capacity is enabled.

DESCRIPTION OF DRAWING(S) - The figure shows the relationship between instruction component and class file information and relationship between the two. (Drawing includes non-English language text).

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

Full	Draw Desc	Clip Img	Image
------	-----------	----------	-------

☐ 10. Document ID: EP 1128263 A2 JP 2001312406 A CN 1312502 A

L16: Entry 10 of 18

File: DWPI

Aug 29, 2001

DERWENT-ACC-NO: 2002-165725

DERWENT-WEEK: 200222

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Program generation apparatus for program execution system, replaces variable name in each class file by assigned offset number so that same offset numbers are assigned to non-dependent variables with same variable name

INVENTOR: KAWAI, M; KAWAMOTO, T

PRIORITY-DATA: 2000JP-0043499 (February 21, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
EP 1128263 A2	August 29, 2001	E	074	G06F009/44
JP 2001312406 A	November 9, 2001		036	G06F009/44
CN 1312502 A	September 12, 2001		000	G06F009/45

INT-CL (IPC): G06 F 9/44; G06 F 9/45

ABSTRACTED-PUB-NO: EP 1128263A

BASIC-ABSTRACT:

NOVELTY - A storage unit prestores class files (52,53) for each terminal apparatus (20). Each class defines dependent variables and non-dependent variables. Light-weight class file (54) for each terminal apparatus, is generated by replacing each variable name in each class file with an assigned offset number such that same offset numbers are assigned to non-dependent variables having same variable name.

DETAILED DESCRIPTION INDEPENDENT CLAIMS are also included for the following:

- (a) Virtual machine;
- (b) Program generation method;
- (c) Computer readable record medium storing program for generating light-weight class files;
- (d) Program execution method

USE - For generating execution programs by compiling and linking source programs written in Java or object oriented language in program execution system.

ADVANTAGE - Since different offset numbers are generated for each variable defined in class files, same offset numbers are assigned to non-dependent variables having same variable names in different terminal apparatuses, when linkage is performed and hence by compiling and linking a common program, light-weight class files are generated and the necessity of linkage for each terminal apparatus is eliminated.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram showing the construction of a program execution system.

Terminal apparatus 20

Class files 52-54

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

RMC	Draw Desc	Clip Img	Image
-----	-----------	----------	-------

☐ 11. Document ID: JP 2001051853 A

L16: Entry 11 of 18

File: DWPI

Feb 23, 2001

DERWENT-ACC-NO: 2001-262097

DERWENT-WEEK: 200127

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Programming interface system for object oriented software designing, sets identification number of each programming interface and method based on which character row representing execution program is output

PRIORITY-DATA: 1999JP-0224450 (August 6, 1999)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
JP 2001051853 A	February 23, 2001		019	G06F009/45

INT-CL (IPC): G06 F 9/44; G06 F 9/45

ABSTRACTED-PUB-NO: JP2001051853A

BASIC-ABSTRACT:

NOVELTY - Java language class files from source programs (A11-E15) are interlinked with static link program (100) to generate executable program (21). Identification number relative to each program interface and number relative to programming method are set by setting units (101,102). Character row representing execution program is generated by replacing interface and method number suitably using converter (105).

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for virtual program execution machine.

USE - For software designing using object oriented language e.g. Java.

ADVANTAGE - Since character row is substituted by interface and method number, size of program is small, hence load of execution is reduced.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of programming interface system. (Drawing includes non-English language text).

Executable program 21

Static link program 100

Setting units 101,102

Converter 105

Source programs A11-E15

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

DOC	Draw Desc	Clip Img	Image
-----	-----------	----------	-------

☐ 12. Document ID: US 6151701 A

L16: Entry 12 of 18

File: DWPI

Nov 21, 2000

DERWENT-ACC-NO: 2001-181304

DERWENT-WEEK: 200118

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Decompiler used for program debugging system, executes decompiled file and generates number map and symbol table corresponding to decompiled executable file

INVENTOR: HUMPHREYS, G; MARTINO, P J

PRIORITY-DATA: 1997US-060480P (September 30, 1997), 1998US-0162472 (September 28, 1998)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 6151701 A	November 21, 2000		013	G06F009/45

INT-CL (IPC): G06 F 9/45

ABSTRACTED-PUB-NO: US 6151701A

BASIC-ABSTRACT:

NOVELTY - A compiler (32) receives source code file (34) and produces executable file (30). The executable file is then decompiled by a decompiler (40) corresponding to which a table generator (44) generates a line number map (48) and a symbol table (50). The table generator is operated by hooks (42) connected to the decompiler.

USE - For program debugging system.

ADVANTAGE - Programmer can easily debug a class file or other programs without having to worry about recompilation.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of debugging system.

Files 30,34

Compiler 32

Decompiler 40

Hooks 42

Table generator 44

Line number map 48

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

RMK	Draw Desc	Clip Img	Image
-----	-----------	----------	-------

☐ 13. Document ID: RD 431147 A

L16: Entry 13 of 18

File: DWPI

Mar 10, 2000

DERWENT-ACC-NO: 2000-348279

DERWENT-WEEK: 200030

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Testing a Java (RTM) application having a jar file containing class files by taking the jar file, running tests and automatically indicating the coverage of the tests.

PRIORITY-DATA: 2000RD-0431147 (February 20, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
RD 431147 A	March 10, 2000		001	G06F000/00

INT-CL (IPC): G06 F 0/00

ABSTRACTED-PUB-NO: RD 431147A

BASIC-ABSTRACT:

NOVELTY - Using unjar provides a list of the classes in the program code. Chopping off the .class at the end of each line lists the classes which need to be tested and the tests are then performed using the -verbose option which profiles all the classes. Going through the list of files looking for the class in the profile output provides a count of the number of classes which have been tested and those which have been missed.

USE - Testing Java (RTM) applications.

ADVANTAGE - No recompilation is needed because the arrangement operates on the compiled code thus providing an automatic link between the program code and the test results.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

RMK	Draw Desc	Image
-----	-----------	-------

☐ 14. Document ID: EP 905617 A2 US 6233733 B1 CA 2249042 A1 JP 11242597 A

L16: Entry 14 of 18

File: DWPI

Mar 31, 1999

DERWENT-ACC-NO: 1999-192821

DERWENT-WEEK: 200129

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Byte code linking method in uninterrupted block of byte codes in formation of data flow graph

INVENTOR: GHOSH, S

PRIORITY-DATA: 1997US-0940212 (September 30, 1997)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
EP 905617 A2	March 31, 1999	E	011	G06F009/45
US 6233733 B1	May 15, 2001		000	G06F009/45
CA 2249042 A1	March 30, 1999	E	000	G06F009/45
JP 11242597 A	September 7, 1999		033	G06F009/45

INT-CL (IPC): G06 F 9/45

ABSTRACTED-PUB-NO: EP 905617A
BASIC-ABSTRACT:

NOVELTY - The method involves forward scanning the block of byte codes to identify a start of each of the byte codes, and backward scanning each of the byte codes. A link is generate in the data flow graph that links each f the byte codes to all other byte codes used y each of the byte codes.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is included for a computer readable storage medium.

USE - For linking byte codes in uninterrupted block of byte codes in formation of data flow graph for optimization of Java class files

ADVANTAGE - Optimizes Java class files by improving efficiency of generating a data flow graph for the intermediate representation.

DESCRIPTION OF DRAWING(S) - The figure shows a flow diagram of the optimization of a JAVA byte code source file suing stand alone optimizer.
ABSTRACTED-PUB-NO:

US 6233733B EQUIVALENT-ABSTRACTS:

NOVELTY - The method involves forward scanning the block of byte codes to identify a start of each of the byte codes, and backward scanning each of the byte codes. A link is generate in the data flow graph that links each f the byte codes to all other byte codes used y each of the byte codes.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is included for a computer readable storage medium.

USE - For linking byte codes in uninterrupted block of byte codes in formation of data flow graph for optimization of Java class files

ADVANTAGE - Optimizes Java class files by improving efficiency of generating a data flow graph for the intermediate representation.

DESCRIPTION OF DRAWING(S) - The figure shows a flow diagram of the optimization of a JAVA byte code source file suing stand alone optimizer.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

Home	Draw Desc	Clip Img	Image
------	-----------	----------	-------

☐ 15. Document ID: US 5854896 A

L16: Entry 15 of 18

File: DWPI

Dec 29, 1998

DERWENT-ACC-NO: 1999-095170

DERWENT-WEEK: 199908

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Persistent logical partitions preservation method for use in distributed parallel processing system environment - involves creating special configuration object class file in system data repository which is used for mapping nodes to their respective sub-environments

INVENTOR: BRENNER, L B; BRISKEY, K C ; ROTHAUPT, K K

PRIORITY-DATA: 1996US 564577 (June 17, 1996)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 5854896 A	December 29, 1998		011	G06F013/00

INT-CL (IPC): G06 F 13/00

ABSTRACTED-PUB-NO: US 5854896A

BASIC-ABSTRACT:

The method involves controlling and maintaining all communication and status information between nodes through a central control element (150) connected to the nodes through network (140). All system data is stored in a system data repository located in central control element.

A special configuration object class file is created in system data repository using the stored system data. The object class file is used for mapping nodes to their respective sub-environments. A suitable portion of system data relating to configuration is retrieved and provided to each mode after rebooting completion.

USE - In UNIX based environment.

ADVANTAGE - Preserves integrity of sub-environment after rebooting. Enables mapping of data belonging to each respective sub-environment in respective portion of adapter. Provides address of adapter in destination file to aid restoration of sub-environments after shut down.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

RWC	Draw Desc	Clip Img	Image
-----	-----------	----------	-------

☐ 16. Document ID: DE 69627926 E EP 735474 A2 CA 2172423 A EP 735474 A3 JP 09054685 A EP 735474 B1

L16: Entry 16 of 18

File: DWPI

Jun 12, 2003

DERWENT-ACC-NO: 1996-435867

DERWENT-WEEK: 200346

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Distributed object creation for distributed object system - has wrapper classes in distributed objects inheriting object attributes via inheritance relationship with developer written servant class of objects, classes inherit attributes through optional relationships with interface class

INVENTOR: BALICK, M; BRACHO, R ; HAPNER, M W ; MCCHESENEY, R J ; SNYDER, A ; VAN HOFF, A A

PRIORITY-DATA: 1995US-0414240 (March 31, 1995)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
DE 69627926 E	June 12, 2003		000	G06F009/46
EP 735474 A2	October 2, 1996	E	028	G06F009/46
CA 2172423 A	October 1, 1996		000	G06F009/44
EP 735474 A3	January 15, 1997		000	G06F009/46
JP 09054685 A	February 25, 1997		021	G06F009/06
EP 735474 B1	May 7, 2003	E	000	G06F009/46

INT-CL (IPC): G06 F 9/06; G06 F 9/44; G06 F 9/46

ABSTRACTED-PUB-NO: EP 735474A

BASIC-ABSTRACT:

The method involves providing interface files describing interfaces contained in the distributed object, and implementation files describing object implementations. Servant class files describe a servant class of object implementations. The interface, under computer control, compiles the files to produce compiled header and source files corresp. to the files.

The corresp. implementation files include template source files. Under computer control the header and source files are linked with a wrapper class containing functions and services for operating the distributed object on the system, so that the wrapper class has an inheritance relationship w.r.t. the servant class. The wrapper class is derived from the servant class to create the distributed object.

ADVANTAGE - Can be installed without extensive modification to objects and without programmer having familiarity with support provision for distributed object systems in programming code of their objects.

Full Title Citation Front Review Classification Date Reference Sequences Attachments

RMIC Draw Desc Clip Img Image

☐ 17. Document ID: EP 629962 A2 US 5495603 A EP 629962 A3

L16: Entry 17 of 18

File: DWPI

Dec 21, 1994

DERWENT-ACC-NO: 1995-024432

DERWENT-WEEK: 199614

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Data file classification method for large database systems - using automatic class selection filter including rule declarations specifying values of attributes tested in sequence to establish identification for assignment

INVENTOR: FRUCHTMAN, B; KACZMARSKI, M A ; WALDO, E J

PRIORITY-DATA: 1993US-0077222 (June 14, 1993)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
EP 629962 A2	December 21, 1994	E	009	G06F015/403
US 5495603 A	February 27, 1996		008	G06F017/30
EP 629962 A3	April 5, 1995		000	G06F015/403

INT-CL (IPC): G06F 15/403; G06F 17/30

ABSTRACTED-PUB-NO: EP 629962A

BASIC-ABSTRACT:

The method includes establishing at least one Automatic Class Selection filter (ACS) (28) in the storage arrangement, such that the filter includes an ordered multiple of rule declarations. Each declaration specifies a range of values for multiple file attributes and at least one file management class. Each rule declaration is tested in sequence for coincidence with the file attributes of a first data file (22) until identification is established.

The file management class specified in the first coincident rule declaration is then assigned (24) to the first data file. The file attributes include multiple file identifiers and file characteristics. Each rule declaration specifies a value for at least one of the file characteristics. The assigned file management classification is stored. Identifiers include a file name, a file directory and a file storage group.

USE/ADVANTAGE - For e.g. Multiple Virtual Storage and Systems Managed Storage systems. Requires no programming skills to specify the data file class selection criteria. File newly selected whenever data file presented to filter. Dynamic updating of data file class linkage. Dynamic temporal data file class selection. Independent of platform. Highly optimised matching procedure. (Reissued from week 9504 to add classifications/ Printed in week 9511)

US 5495603A EQUIVALENT-ABSTRACTS:

In a computer system having means for storing a plurality of files each having one or more file attributes and for storing at least one Automatic Class Selection (ACS) filter, a method for assessing the file management class of said files comprising the steps of:

responsive to the opening of a first said file, referencing said Automatic Class Selection (ACS) filter in said storage means wherein said ACS filter includes an ordered plurality of rule declarations each specifying a range of values for a plurality of said file attributes and at least one file management class;

testing in sequence each said rule declaration for coincidence with the file attributes of said first file until first encountering a rule declaration that coincides with said first file attributes; and

selecting for said first file the file management class specified in said first encountered coincident rule declaration; and

retesting said rule declarations and reselecting said file management class for said first file responsive to a subsequent reopening of said first file.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

Full	Draw Desc	Clip Img	Image
------	-----------	----------	-------

☐ 18. Document ID: EP 476841 A US 5317728 A

L16: Entry 18 of 18

File: DWPI

Mar 25, 1992

DERWENT-ACC-NO: 1992-098519

DERWENT-WEEK: 199213

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Storage management across file systems - uses repository file space to accommodate storage of management classes for files of second file system for which empty file os created

INVENTOR: TEVIS, G J; WALDO, E ; WALDO, E J

PRIORITY-DATA: 1990US-0578386 (September 7, 1990)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
EP 476841 A	March 25, 1992		022	
US 5317728 A	May 31, 1994		022	G06F015/40

INT-CL (IPC): G06F 15/40

ABSTRACTED-PUB-NO: EP 476841A

BASIC-ABSTRACT:

The data processing system includes a host computer and a number of peripheral data storage arranged in a data storage hierarchy, and comprises a level 0 storage (14) including two storage spaces organized according to two file systems. The first storage space includes a catalogue of the first management information storage management of the files on the second storage space.

A host processor (10) is coupled to the level 0 and a level 1 storage (15). A management unit (24) is coupled to the host processor for instructing the host processor to manage the files on the second storage space using the first management information. The second storage space includes a catalogue which has no available storage space for storage of the first management information.

ADVANTAGE - Provides different solution to problem of catalogue enlargement which

normally results in restructuring and reprogramming of operating systems and which has severe impact on existing applications expecting precise current location of catalogue.

ABSTRACTED-PUB-NO:

US 5317728A EQUIVALENT-ABSTRACTS:

A repository file space in the first file system is used to accomplish the storage of some or all of the management classes or attributes for the files of the second file system. For each file in the second file system an empty file is created in the repository file space of the first file system. The creation of an empty file in the repository file space causes a management class to be selected for the file and stored in the catalog of the first file system as would be done for any file in the first file system.

A naming convention is used to link each file in the repository file space with its associated file in the second file system. The additional files in the first file system are left empty as no other information need be stored for storage management purposes. The management attributes of the files are then compared to the management criteria represented by the management classes stored in the catalog of the first file system and the files are managed accordingly.

USE/ADVANTAGE - In storage management of both mini-disc and shared file system in virtual machine operating system environment. Provision for data back-up within data storage hierarchy and exclusion manual operation.

Full Title Citation Front Review Classification Date Reference Sequences Attachments

RVC Draw Desc Clip Img Image

Generate Collection

Print

Terms	Documents
L15	18

Display Format:

-

Change Format

[Previous Page](#)

[Next Page](#)

Generate Collection

Print

OS search record Search Results - Record(s) 1 through 11 of 11 returned.

☐ 1. Document ID: US 20020170047 A1

L18: Entry 1 of 11

File: PGPB

Nov 14, 2002

PGPUB-DOCUMENT-NUMBER: 20020170047
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020170047 A1

TITLE: System and method for transforming object code

PUBLICATION-DATE: November 14, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Swetland, Brian	Mountain View	CA	US	

US-CL-CURRENT: 717/162

ABSTRACT:

A unified programming object is described comprising: a shared constant pool comprising global constant pool entries mapped from local constant pool entries of two or more class files; and a plurality of object code copied from the two or more class files to the unified programming object and identified by the global constant pool entries.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	RVMC	Draw Desc	Image
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	-----------	-------

☐ 2. Document ID: US 20010047513 A1

L18: Entry 2 of 11

File: PGPB

Nov 29, 2001

PGPUB-DOCUMENT-NUMBER: 20010047513
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010047513 A1

TITLE: Computer program product having preloaded software module

PUBLICATION-DATE: November 29, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Tock, Theron D.	Sunnyvale	CA	US	

US-CL-CURRENT: 717/162

ABSTRACT:

A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein

described. The executable module represents Java classes that are structured for dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	EMC	Draw Desc	Image
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	-----	-----------	-------

☐ 3. Document ID: US 6618769 B1

L18: Entry 3 of 11

File: USPT

Sep 9, 2003

US-PAT-NO: 6618769

DOCUMENT-IDENTIFIER: US 6618769 B1

TITLE: Module-by-module verification

DATE-ISSUED: September 9, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bracha; Gilad	Los Altos	CA		
Liang; Sheng	Mountain View	CA		
Lindholm; Timothy G.	Palo Alto	CA		

US-CL-CURRENT: 709/332; 717/165

ABSTRACT:

A method, computer program, signal transmission and apparatus pre-verify instructions in a module of a computer program one module-at-a-time. First it is determined whether checking an instruction in a first module which is loaded requires information in a referenced module different than the first module. If the information is required, a constraint for the referenced module is written without loading or otherwise accessing the referenced module. During linking it is determined whether a first module which is loaded has passed pre-verification one-module-at-a-time before linking. A pre-verification constraint on a constrained module is read, if any, if the first module has passed such verification. If any pre-verification constraint is read, the pre-verification constraint is enforced if the constrained module is already loaded.

21 Claims, 20 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 18

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	EMC	Draw Desc	Image
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	-----	-----------	-------

☐ 4. Document ID: US 6530080 B2

L18: Entry 4 of 11

File: USPT

Mar 4, 2003

US-PAT-NO: 6530080

DOCUMENT-IDENTIFIER: US 6530080 B2

TITLE: Method and apparatus for pre-processing and packaging class files

DATE-ISSUED: March 4, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Fresko; Nedim	San Francisco	CA		
Tuck; Richard	San Francisco	CA		

US-CL-CURRENT: 717/166; 717/143

ABSTRACT:

A method and apparatus for pre-processing and packaging class files. Embodiments remove duplicate information elements from a set of class files to reduce the size of individual class files and to prevent redundant resolution of the information elements. Memory allocation requirements are determined in advance for the set of classes as a whole to reduce the complexity of memory allocation when the set of classes are loaded. The class files are stored in a single package for efficient storage, transfer and processing as a unit. In an embodiment, a pre-processor examines each class file in a set of class files to locate duplicate information in the form of redundant constants contained in a constant pool. The duplicate constant is placed in a separate shared table, and all occurrences of the constant are removed from the respective constant pools of the individual class files. During pre-processing, memory allocation requirements are determined for each class file, and used to determine a total allocation requirement for the set of class files. The shared table, the memory allocation requirements and the reduced class files are packaged as a unit in a multi-class file.

45 Claims, 6 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 6

Full Title Citation Front Review Classification Date Reference Sequences Attachments

PMW Draw Desc Image

☐ 5. Document ID: US 6446254 B1

L18: Entry 5 of 11

File: USPT

Sep 3, 2002

US-PAT-NO: 6446254

DOCUMENT-IDENTIFIER: US 6446254 B1

TITLE: Packaging memory image files

DATE-ISSUED: September 3, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chapman; Graham	Ottawa			CA
Duimovich; John	Ottawa			CA
Gray-Donald; Trent	Ottawa			CA
Johnson; Graeme	Ottawa			CA
Low; Andrew	Ottawa			CA
Burka; Peter Wiebe	Ottawa			CA
Mueller; Patrick James	Apex	NC		
Sciampacone; Ryan Andrew	Ottawa			CA
Shipton; Peter Duncan	Ottawa			CA

US-CL-CURRENT: 717/116; 717/118, 717/165

ABSTRACT:

In typical Java and other interpreted programming language environments, the code is stored in ROM in a semi-processed state, class files containing byte codes. When the device is turned on, a virtual machine resolves references and links the class file in RAM to permit desired applications to be run. In the invention, the class files are further pre-processed to select the data which will not change or require updating. This data is packaged into memory image files containing internal data pre-linking this data. The memory image files are stored in ROM and are accessible from ROM by the virtual machine at runtime. Only elements that will be updated, such as the objects themselves, must be instantiated in RAM at runtime. This reduces the amount of RAM needed to run the application. In an environment with memory constraints, the reduction in RAM requirements permits more RAM to be made available for application use.

11 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 8

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

ROMC	Draw Desc	Image
------	-----------	-------

☐ 6. Document ID: US 6339841 B1

L18: Entry 6 of 11

File: USPT

Jan 15, 2002

US-PAT-NO: 6339841
DOCUMENT-IDENTIFIER: US 6339841 B1

TITLE: Class loading model

DATE-ISSUED: January 15, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Merrick; Roland Albert	Harvington			GB
Webb; Alan Michael	Chandlers Ford			GB

US-CL-CURRENT: 717/166; 707/103R, 709/203, 717/118

ABSTRACT:

This invention relates to a method of loading Java ClassFiles on to a Java Virtual Machine. On a regular JVM the ClassFile are loaded as and when required. In this specification there is described a method of implementing an object oriented program language such as Java on a computer. The method comprises identifying a class, one of the basic building blocks of the language, which is not within the program domain, that is not loaded into the Java a Virtual Machine. Next it introduces to the program domain only the minimum components of the class which are necessary for commencing processing of the class. The class may comprise several blocks of data representing the methods of the class, since the class may only have been identified because one of the methods within the class was referenced then only the block of data representing this method is loaded into the Java Virtual Machine along with the other essential components of the class. Other blocks of data representing methods can be loaded as and when required by the programming domain. Redundant method components may be removed from the program domain to save memory.

22 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

ROMC	Draw Desc	Image
------	-----------	-------

☐ 7. Document ID: US 6223346 B1

L18: Entry 7 of 11

File: USPT

Apr 24, 2001

US-PAT-NO: 6223346

DOCUMENT-IDENTIFIER: US 6223346 B1

TITLE: Computer program product having preloaded software module

DATE-ISSUED: April 24, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tock; Theron D.	Sunnyvale	CA		

US-CL-CURRENT: 717/166; 713/1, 713/2

ABSTRACT:

A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein described. The executable module represents Java classes that are structured for dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

7 Claims, 14 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 12

Full Title Citation Front Review Classification Date Reference Sequences Attachments

Full Draw Desc Image

☐ 8. Document ID: US 6202208 B1

L18: Entry 8 of 11

File: USPT

Mar 13, 2001

US-PAT-NO: 6202208

DOCUMENT-IDENTIFIER: US 6202208 B1

**** See image for Certificate of Correction ****

TITLE: Patching environment for modifying a Java virtual machine and method

DATE-ISSUED: March 13, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Holiday, Jr.; Matthew R.	Allen	TX		

US-CL-CURRENT: 717/166; 707/103R, 711/6, 717/168

ABSTRACT:

The invention includes a patch environment for a modifying a program executed by a Java Virtual Machine ("JVM") while the program is being executed. The patch

environment has a patch data structure defined on an electronic memory of the computer. The patch data structure has at least one Java patch for modifying a loader environment of the JVM. A plurality of data items contained in a data structure defined on the electronic memory of the computer represents each patch of the patch data structure. A second data item is contained in a second data structure defined on the electronic memory of the computer, the data item representing each applied patch of the patch data structure that modifies the loader environment of the JVM. The method of the present invention applies an ordered set of changes to a Java program while running under the control of a Java Virtual Machine having a loader environment which manages the program's loaded software. A patch environment is created such that the patch environment can alter the loader environment of the JVM. A patch file is generated containing a change to be applied to the loaded software and is loaded into the patch environment. The patch is then applied to the loaded software by changing the loader environment of the JVM while the Java program is running.

12 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

RPMC	Draw Desc	Image
------	-----------	-------

☐ 9. Document ID: US 5987256 A

L18: Entry 9 of 11

File: USPT

Nov 16, 1999

US-PAT-NO: 5987256
DOCUMENT-IDENTIFIER: US 5987256 A

TITLE: System and process for object rendering on thin client platforms

DATE-ISSUED: November 16, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wu; Bo	San Jose	CA		
Lu; Ling	San Jose	CA		

US-CL-CURRENT: 717/146; 717/118, 717/162

ABSTRACT:

A system for processing an object specified by an object specifying language such as HTML, JAVA or other languages relying on relative positioning, that require a rendering program utilizing a minimum set of resources, translates the code for use in a target device that has limited processing resources unsuited for storage and execution of the HTML rendering program, JAVA virtual machine, or other rendering engine for the standard. Data concerning such an object is generated by a process that includes first receiving a data set specifying the object according to the object specifying language, translating the first data set into a second data set in an intermediate object language adapted for a second rendering program suitable for rendering by the target device that utilizes actual target display coordinates. The second data set is stored in a machine readable storage device, for later retrieval and execution by the thin client platform.

14 Claims, 15 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 11

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

RPMC	Draw Desc	Image
------	-----------	-------

☐ 10. Document ID: US 5966542 A

L18: Entry 10 of 11

File: USPT

Oct 12, 1999

US-PAT-NO: 5966542

DOCUMENT-IDENTIFIER: US 5966542 A

TITLE: Method and system for loading classes in read-only memory

DATE-ISSUED: October 12, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tock; Theron D.	Sunnyvale	CA		

US-CL-CURRENT: 717/166; 713/1, 713/2, 717/116, 717/163

ABSTRACT:

A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein described. The executable module represents Java classes that are structured for dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

6 Claims, 13 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 12

Full Title Citation Front Review Classification Date Reference Sequences Attachments

KMIC Draw Desc Image

☐ 11. Document ID: US 5815718 A

L18: Entry 11 of 11

File: USPT

Sep 29, 1998

US-PAT-NO: 5815718

DOCUMENT-IDENTIFIER: US 5815718 A

TITLE: Method and system for loading classes in read-only memory

DATE-ISSUED: September 29, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tock; Theron D.	Sunnyvale	CA		

US-CL-CURRENT: 717/166; 709/331, 717/118

ABSTRACT:

A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein described. The executable module represents Java classes that are structured for

dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

20 Claims, 14 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 12

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

RVNC	Draw Desc	Image
------	-----------	-------

Generate Collection

Print

Terms	Documents
L8 and l1	11

Display Format:

-

Change Format

[Previous Page](#)

[Next Page](#)

Welcome to IEEE Xplore®

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Your search matched **2** of **978562** documents.

A maximum of **2** results are displayed, **15** to a page, sorted by **Relevance** in **descending** order.

You may refine your search by editing the current search expression or entering a new one in the text box.

Then click **Search Again**.

(constant pool)

Search Again

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

Print Format

Results:

Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD**

1 A study of code reuse and sharing characteristics of Java applications

Conte, M.T.; Trick, A.R.; Gyllenhaal, J.C.; Hwu, W.W.;

Workload Characterization: Methodology and Case Studies, 1998 , 29 Nov. 1998

Page(s): 27 -35

[\[Abstract\]](#) [\[PDF Full-Text \(112 KB\)\]](#) **IEEE CNF**

2 Jato: a compact binary file format for Java class

Sheng-De Wang; Lin, Y.;

Parallel and Distributed Systems, 2001. ICPADS 2001. Proceedings. Eighth International Conference on , 26-29 June 2001

Page(s): 467 -474

[\[Abstract\]](#) [\[PDF Full-Text \(556 KB\)\]](#) **IEEE CNF**





Try the *new* Portal design
Give us your opinion after using it.

Search Results




Search Results for: [bytecode<AND>((redundant<AND>((class file<AND>((constant pool))))))]
Found 18 of 121,820 searched.

Search within Results

 [> Advanced Search](#)
[> Search Help/Tips](#)

Sort by: [Title](#) [Publication](#) [Publication Date](#) [Score](#)  [Binder](#)

Results 1 - 18 of 18 [short listing](#)

- 1 [Practical experience with an application extractor for Java](#) 89%
 Frank Tip , Chris Laffra , Peter F. Sweeney , David Streeter
ACM SIGPLAN Notices , Proceedings of the 1999 ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications October 1999
Volume 34 Issue 10
Java programs are routinely transmitted over low-bandwidth network connections as compressed class file archives (i.e., zip files and jar files). Since archive size is directly proportional to download time, it is desirable for applications to be as small as possible. This paper is concerned with the use of program transformations such as removal of dead methods and fields, inlining of method calls, and simplification of the class hierarchy for reducing application size. Such “extract ...
- 2 [Practical extraction techniques for Java](#) 88%
 Frank Tip , Peter F. Sweeney , Chris Laffra , Aldo Eisma , David Streeter
ACM Transactions on Programming Languages and Systems (TOPLAS) November 2002
Volume 24 Issue 6
Reducing application size is important for software that is distributed via the internet, in order to keep download times manageable, and in the domain of embedded systems, where applications are often stored in (Read-Only or Flash) memory. This paper explores extraction techniques such as the removal of unreachable methods and redundant fields, inlining of method calls, and transformation of the class hierarchy for reducing application size. We implemented a number of extraction techniques in < ...
- 3 [Techniques for obtaining high performance in Java programs](#) 84%
 Iffat H. Kazi , Howard H. Chen , Berdenia Stanley , David J. Lilja
ACM Computing Surveys (CSUR) September 2000
Volume 32 Issue 3
This survey describes research directions in techniques to improve the performance of programs

written in the Java programming language. The standard technique for Java execution is interpretation, which provides for extensive portability of programs. A Java interpreter dynamically executes Java bytecodes, which comprise the instruction set of the Java Virtual Machine (JVM). Execution time performance of Java programs can be improved through compilation, possibly at the expense of portability ...

4 Soot - a Java bytecode optimization framework

82%

4 Raja Vallée-Rai , Phong Co , Etienne Gagnon , Laurie Hendren , Patrick Lam , Vijay Sundaresan

Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research November 1999

This paper presents Soot, a framework for optimizing Java bytecode. The framework is implemented in Java and supports three intermediate representations for representing Java bytecode: Baf, a streamlined representation of bytecode which is simple to manipulate; Jimple, a typed 3-address intermediate representation suitable for optimization; and Grimp, an aggregated version of Jimple suitable for decompilation. We describe the motivation for each representation, and the salient points in translation ...

5 JAZZ: an efficient compressed format for Java archive files

82%

4 Quetzalcoatl Bradley , R. Nigel Horspool , Jan Vitek

Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research November 1998

The Jazz file format is intended to be a replacement for the JAR file format when used for storage and distribution of Java programs. A Jazz file is compressed to a degree that far exceeds what is possible with a JAR file. The smaller size of the Jazz format permits faster transmission speeds over a network and has the additional benefit of conserving disk storage. The compression is achieved as a combination of different data compression methods, adapted to suit the characteristics of collection ...

6 Software techniques for program compaction: Extracting library-based Java applications

80%

4 Frank Tip , Peter F. Sweeney , Chris Laffra

Communications of the ACM August 2003

Volume 46 Issue 8

Reducing the size of Java applications by creating an application extractor.

7 A framework for optimizing Java using attributes

80%

4 Patrice Pominville , Feng Qian , Raja Vallée-Rai , Laurie Hendren , Clark Verbrugge

Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research November 2000

This paper presents a framework for supporting the optimization of Java programs using attributes in Java class files. We show how class file attributes may be used to convey both optimization opportunities and profile information to a variety of Java virtual machines including ahead-of-time compilers and just-in-time compilers. We present our work in the context of Soot, a framework that supports the analysis and transformation of Java bytecode (class files)[21, 25, 26]. We demonstrate the framework ...

8 Engineering a customizable intermediate representation

77%

4 K. Palacz , J. Baker , C. Flack , C. Grothoff , H. Yamauchi , J. Vitek

Proceedings of the 2003 workshop on Interpreters, Virtual Machines and Emulators June 2003

The Ovm framework is a set of tools and components for building language runtimes. We present the intermediate representation and software design patterns used throughout the framework. One of the main themes in this work has been to support experimentation with new linguistic constructs and implementation techniques. To this end, framework components were designed to be parametric with respect to the instruction set on which they operate. We argue that our approach eases the task of writing new ...

9 Language-specific make technology for the Java programming language

77%

4 Mikhail Dmitriev

ACM SIGPLAN Notices , Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications November 2002

Volume 37 Issue 11

Keeping the code of a Java application consistent (code is consistent if all of the project classes can be recompiled together without errors) prevents late linking errors, and thus may significantly improve development turnaround time. In this paper we describe a make technology for the Java programming language, that is based on smart dependency checking, guarantees consistency of the project code, and at the same time reduces the number of source code recompilations to the minimum. After proj ...

10 The apprentice challenge

77%

4 J. Strother Moore , George Porter

ACM Transactions on Programming Languages and Systems (TOPLAS) May 2002

Volume 24 Issue 3

We describe a mechanically checked proof of a property of a small system of Java programs involving an unbounded number of threads and synchronization, via monitors. We adopt the output of the javac compiler as the semantics and verify the system at the bytecode level under an operational semantics for the JVM. We assume a sequentially consistent memory model and atomicity at the bytecode level. Our operational semantics is expressed in ACL2, a Lisp-based logic of recursive functions. Our proofs ...

11 Using annotations to reduce dynamic optimization time

77%

4 Chandra Krintz , Brad Calder

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation May 2001

Volume 36 Issue 5

Dynamic compilation and optimization are widely used in heterogenous computing environments, in which an intermediate form of the code is compiled to native code during execution. An important trade off exists between the amount of time spent dynamically optimizing the program and the running time of the program. The time to perform dynamic optimizations can cause significant delays during execution and also prohibit performance gains that result from more complex optimization.

12 SafeTSA: a type safe and referentially secure mobile-code representation based on static single

77%

4 assignment form

13 Extracting library-based object-oriented applications

77%


 Peter F. Sweeney , Frank Tip

**ACM SIGSOFT Software Engineering Notes , Proceedings of the 8th ACM SIGSOFT
international symposium on Foundations of software engineering: twenty-first century
applications November 2000**
Volume 25 Issue 6

In an increasingly popular model of software distribution, software is developed in one computing environment and deployed in other environments by transfer over the internet. Extraction tools perform a static whole-program analysis to determine unused functionality in applications in order to reduce the time required to download applications. We have identified a number of scenarios where extraction tools require information beyond what can be inferred through static analysis: software distr ...

14 Designing robust Java programs with exceptions

77%


 Martin P. Robillard , Gail C. Murphy

**ACM SIGSOFT Software Engineering Notes , Proceedings of the 8th ACM SIGSOFT
international symposium on Foundations of software engineering: twenty-first century
applications November 2000**
Volume 25 Issue 6

Exception handling mechanisms are intended to help developers build robust systems. Although an exception handling mechanism provides a basis for structuring source code dealing with unusual situations, little information is available to help guide a developer in the appropriate application of the mechanism. In our experience, this lack of guidance leads to complex exception structures. In this paper, we reflect upon our experiences using the Java exception handling mechanism. Based on these ...

15 Using production grammars in software testing

77%


 Emin Gün Sirer , Brian N. Bershad

**ACM SIGPLAN Notices , Proceedings of the 2nd conference on Domain-specific
languages December 1999**
Volume 35 Issue 1

Extensible typesafe systems, such as Java, rely critically on a large and complex software base for their overall protection and integrity, and are therefore difficult to test and verify. Traditional testing techniques, such as manual test generation and formal verification, are too time consuming, expensive, and imprecise, or work only on abstract models of the implementation and are too simplistic. Consequently, commercial virtual machines deployed so far have exhibited numerous bugs and ...

16 Compressing Java class files

77%

 William Pugh

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on


Programming language design and implementation May 1999

Volume 34 Issue 5

Java class files are often distributed as jar files, which are collections of individually compressed class files (and possibly other files). Jar files are typically about 1/2 the size of the original class files due to compression. I have developed a wire-code format for collections of Java class files. This format is typically 1/2 to 1/5 of the size of the corresponding compressed jar file (1/4 to 1/10 the size of the original class files).

17 Software watermarking: models and dynamic embeddings

77%

 Christian Collberg , Clark Thomborson

Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages January 1999

18 Manufacturing cheap, resilient, and stealthy opaque constructs

77%

 Christian Collberg , Clark Thomborson , Douglas Low

Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages January 1998

Results 1 - 18 of 18 short listing

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2003 ACM, Inc.